1 Preliminary Topics

1) Exact string matching

    a) Introduction
    b) The Z Algorithm[1]
    c) Knuth-Morris-Pratt and Boyer-Moore
    d) Seminumerical matching: Rabin-Karp & Shift-And

2) Inexact matching (edit distance, alignment in linear space, Four-Russians' speedup). Example: BLAST.

    a) Edit Distance with dynamic programming
    b) Edit distance demo
    c) Hirschberg's algorithm for linear space alignment
    d) General and affine gap penalties for edit distance
    e) Four Russians speedup for edit distance

3) Suffix trees and arrays and their applications; Ukkonen's suffix tree construction algorithm; Burrows-Wheeler transform, Suffix Tree of Alignment: An Efficient Index for Similar Data

    a) Suffix trees
    b) More suffix tree applications
    c) Suffix arrays
    d) Minimizers, sparse suffix arrays
    e) Burrows-Wheeler transform and FM-index

4) Multiple sequence alignment; motif finding; multiple patterns. Example application: Information phylogenies.

    a) Gibbs sampling for motif finding
    b) Multiple sequence alignment
    c) Multiple pattern search

5) Hashing / randomization techniques for big data. Example application: TBD.

    a) Feedback shift registers (FSR) & de Bruijn graphs / de Bruijn sequences
    b) Locality sensitive hashing for strings (Approximating nearest neighbors [ANN])
    c) Nearest neighbor search with locality sensitive hashing
    d) Motif discovery using random projection
    e) Bloom filters for detecting set membership[2] [3]

---

[1] a linear time string matching algorithm which runs in O(n) complexity

[2] The basic bloom filter supports two operations: **test** and **add**.
Test is used to check whether a given element is in the set or not. If it returns:

- *false* then the element is definitely not in the set.
- *true* then the element is *probably* in the set. The *false positive rate* is a function of the bloom filter's size and the number and independence of the hash functions used.

[3] https://pypi.org/project/bloom-filter/

f) Space efficient probabilistic data structures
g) Sequence Bloom Trees

6) Compression and other compressed indices.

   a) Compressing and Indexing Massive data
   b) Wavelet trees for all
   c) Lempel-Ziv parsing and universal lossless compression
   d) Bidirectional Burrows Wheeler transform (BWT)

7) State machines. Example application: Finding Advice Reification in online discussion.

   a) Regular expressions and Finite State Automations (FSAs) [five tuples]
   b) Context-free grammars, parsers (top down and bottom up)
   c) Hidden Markov Models

8) String graphs. Example application: identifying contextual information alignment in a population of communication records.

   a) Structural variants and short tandem repeats using variant graphs
   b) Burrow Wheeler Transforms for labeled trees and Directed Acyclic Graphs (DAG)
   c) Burrow Wheeler transforms for de Bruijn graphs: discovering knowledge resharing.

9) Current research in "Big Social Data".

   a) TBD